REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

0089

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	30 November 2000	Final Technical Report (1 February 1998 - 30 November 2000)

4. TITLE AND SUBTITLE
Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms

5. FUNDING NUMBERS
F49620-98-1-0111

6. AUTHOR(S)
Richard Nance
Subcontract to: Sheldon H. Jacobson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
VPI&SU
340 Burruss Hall
Blacksburg, VA 24061

8. PERFORMING ORGANIZATION REPORT NUMBER
#1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR
801 North Randolph Street, Room 732
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Discrete manufacturing process design optimization (DMPDO) is a problem of significant importance and interest to the Air Force. Moreover, the complexity of parts that must be manufactured for airplane engines and other units also makes this problem extremely difficult, due in part to the large number of design sequences that exist for each given part. This has forced researchers to develop heuristic strategies to address such design problems. This report summarizes the research that has been conducted in developing the generalized hill climbing (GHC) algorithm framework for discrete optimization problems in general, and the DMPDO problems in particular. New necessary and sufficient convergence conditions for GHC algorithms have been developed that use a new algorithm iteration classification scheme. Moreover, new performance measures are formulated that more closely capture how practitioners use GHC algorithms to solve large-scale real-world problems. These results provide fundamental insights into how such algorithms should be applied, as well as provide answers to open questions concerning the convergence and performance of various GHC algorithms. On-going interactions with researchers at the Materials Process Design Branch of the AFRL at WPAFB and at Austral Engineering and Software, Inc. has resulted in new software modules for GHC algorithms that are being used by to solve various DMPDO problems of interest to the Air Force.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Manufacturing Process Design Optimization, Discrete Optimization, Heuristics	35
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
2 Unclassified	Unclassified	Unclassified	UL

FINAL TECHNICAL REPORT

Submitted to

Dr. Neal D. Glassman
AFOSR/NM
801 North Randolph Street, Room 732
Arlington, VA 22203-1977
(703) 696-8431
neal.glassman@afosr.af.mil

Principal Investigator:

Sheldon H. Jacobson
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2906
(217) 244-7275 (office)
(217) 244-6534 (fax)
shj@uiuc.edu
www.staff.uiuc.edu/~shj/shj.html

Grant Number: F49620-98-1-0111

Table of Contents

EXECUTIVE SUMMARY

The research conducted under this grant focused on the development and analysis of the generalized hill climbing algorithm framework as a tool for address intractable discrete optimization problems. The major technical accomplishments achieved include

i) the introduction and development of a new algorithm classification scheme for generalized hill climbing algorithms,

ii) the formulation of new necessary and sufficient conditions for generalized hill climbing algorithms,

iii) the formulation of new performance measure for generalized hill climbing algorithms, that more closely matches how practitioners apply such algorithms.

iv) the formulation of a new necessary convergence condition for generalized hill climbing algorithms using these new performance measures. This condition resulted in a nonconvergence proof for threshold accepting, hence resolving an open question in the literature.

v) the benchmarking of generalized hill climbing algorithms using the new performance measures, resulting in a base convergent rate to assess the value of such algorithms.

In addition, several other problems were studied during the grant period, including estimation procedures and complexity results for discrete event simulation problems, aviation security system design and optimization issues, a combinatorial medical problem, and a flexible assembly system design and scheduling problem. All the accomplishments are documented in several archival journal articles and conference proceedings. In addition, many of the results have been presented at national and international conferences, and have won awards for their contribution.

Two Ph.D. dissertations were completed during the period of the grant. Dr. Kelly Ann Sullivan successfully defended and submitted her Ph.D. dissertation "On the Convergence of Generalized Hill Climbing Algorithms" in May 1999. Dr. Diane E. Vaughan successfully defended and submitted her Ph.D. dissertation "Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problem" in August 2000.

1. Generalized Hill Climbing Algorithms

Generalized hill climbing algorithms (Johnson 1996, Jacobson et al. 1998) provide a well-defined framework to model algorithms for intractable discrete optimization problems. Generalized hill climbing algorithms allow inferior solutions to be visited enroute to optimal solutions. This hill climbing capability is the basis for the search strategy's name. To formally describe the generalized hill climbing algorithm framework, several definitions are needed.

For a discrete optimization problem, define the *solution space*, Ω, as a finite set of all possible solutions. Define an *objective function* $f: \Omega \rightarrow [0,+\infty]$ that assigns a non-negative value to each element of the solution space. An important component of GHC algorithms is the *neighborhood function*, $\eta: \Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$. Solutions in a neighborhood are generated *uniformly* at each iteration of a GHC algorithm execution if, for all $\omega \in \Omega$, with $\omega' \in \eta(\omega)$,

$$P\{\omega' \text{ is selected as the neighbor of } \omega \text{ at a given iteration of a GHC algorithm}\} = 1 / |\eta(\omega)|.$$

Unless otherwise stated, assume that the neighbors are generated uniformly at each iteration of a GHC algorithm. The GHC algorithm is described in pseudo-code form (Jacobson et al. 1998):

Set the outer loop counter bound K and the inner loop counter bounds N(k), k = 1,2,...,K
Define a set of hill climbing (random) variables $R_k: \Omega \times \Omega \rightarrow \Re \cup \{-\infty,+\infty\}$, k = 1,2,...K
Set the iteration indices i = 0, k = n = 1
Select an initial solution $\omega(0) \in \Omega$
Repeat while $k \leq K$
  Repeat while $n \leq N(k)$
    Generate a solution $\omega \in \eta(\omega(i))$ and calculate $\delta(\omega(i),\omega) = f(\omega) - f(\omega(i))$
    If $R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega$ (accept improving or hill climbing moves)
    If $R_k(\omega(i),\omega) < \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega(i)$ (reject hill climbing moves)
    $n \leftarrow n+1, i \leftarrow i+1$
  Until n = N(k)
  $n \leftarrow 1, k \leftarrow k+1$
Until k = K

Note that the outer and inner loop bounds, K and N(k), k = 1,2,...,K, respectively, may all be fixed, or K can be fixed and the N(k), k = 1,2,...,K, are random variables whose values are determined by the solution at the end of each set of inner loop iterations satisfying some property (e.g., the solution is a local optima). Moreover, assume that the hill climbing random variables have finite means and finite variances for all k and for all possible pairs of elements in the solution space (i.e., $E[R_k(\omega(i),\omega)] < +\infty$ and $Var[R_k(\omega(i),\omega)] < +\infty$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and for all k = 1,2,...,K, i = 1,2,...,$I = \sum_{k=1}^{K} N(k)$).

The neighborhood function establishes relationships between the solutions in the solution space, hence allows the solution space to be traversed or searched by moving between solutions. To ensure that

the solution space is not fragmented, assume that all the solutions in the solution space (with neighborhood function η) are *reachable;* that is, for all $\omega', \omega'' \in \Omega$, there exists a set of solutions $\omega_1, \omega_2, \ldots, \omega_m \in \Omega$ such that $\omega_r \in \eta(\omega_{r-1})$, r = 1, 2, ..., m+1, where $\omega' \equiv \omega_0$ and $\omega'' \equiv \omega_{m+1}$. Note that if all solutions in a solution space are reachable, then the solution space (with neighborhood function η) is said to be reachable. The goal is to identify a globally optimal solution ω^* (i.e., $f(\omega^*) \leq f(\omega)$ for all $\omega \in \Omega$).

2. Necessary and Sufficient Convergence Conditions

One accomplishment during the term of this grant has been the development of new necessary and sufficient conditions for the convergence of generalized hill climbing algorithms. Convergence conditions provide one way to assess the value and effectiveness of an algorithm. These results are reported in Sullivan (1999) and Sullivan and Jacobson (2000a). To describe these conditions, several additional definitions are needed.

The objective function, f, and the neighborhood function, η, allow the solution space, Ω, to be decomposed into three mutually exclusive and exhaustive sets:

- a set of *G-local optima*, G (the set of global optima),
- a set of *L-local optima*, $L \equiv L(\eta)$ (the set of local optima that are not G-local optima),
- a set of *hill solutions*, H.

Therefore $G \cup L$ are the set of local optima in Ω associated with neighborhood function η, where by definition, $\Omega = G \cup L \cup H$ with $G \cap L = \varnothing$, $G \cap H = \varnothing$, and $L \cap H = \varnothing$. Note that by definition, for all $\omega \in G$, $\eta(\omega) \cap L = \varnothing$, and for all $\omega \in L$, $\eta(\omega) \cap G = \varnothing$ (i.e., a G-local optimum and a L-local optimum cannot be neighbors of each other).

GHC algorithms can be viewed as sampling procedures over the solution space Ω. The key distinction between different GHC algorithms is in *how* the sampling is performed. For example, Monte Carlo search produces unbiased samples (with replacement) from the solution space, while simulated annealing produces biased samples, guided by the neighborhood function, the objective function, and the temperature parameter. More specifically, simulated annealing can be described as a GHC algorithm by setting $R_k(\omega(i),\omega) = -t_k \ln(\nu_i)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, k = 1,2,...,K, where t_k is the temperature parameter (hence, defines a cooling schedule as $t_k \rightarrow 0$) and $\{\nu_i\}$ are independent and identically distributed U(0,1) random variables (see Johnson 1996, Johnson and Jacobson 2001a,b for a complete discussion on and description of how these algorithms fit into the GHC algorithm framework).

To ensure that the neighborhood function does not fragment the solution space (i.e., decompose the solution space into mutually exclusive subsets of solutions that are not reachable), a neighborhood function η on the solution space Ω is said to have the *local search property* if when local search (i.e.,

$R_k(\omega',\omega'') \equiv 0$ for all $\omega' \in \Omega$, $\omega'' \in \eta(\omega')$, k = 1,2,...,K) is applied with neighborhood function η to the solution space Ω, then for all $\omega_H \in H$, there exists $\omega_{G\cup L} \in G\cup L$ such that the local search algorithm will terminate in $\omega_{G\cup L}$, passing only through elements in H. Assume that all neighborhood functions η associated with a GHC algorithm applied to solution space Ω have the local search property. Note that if a solution space (with neighborhood function η) is reachable, then η also possesses the local search property, though the reverse is not necessarily true.

Each execution of a GHC algorithm generates a sequence (sample) of $I = \sum_{k=1}^{K} N(k)$ solutions. In practice, the best solution obtained over the entire GHC algorithm run, not just the final solution, is reported. This allows the algorithm to aggressively traverse the solution space visiting many inferior solutions enroute to a globally optimal solution, while retaining the best solution obtained throughout the entire GHC run. Each sequence of solutions is a function of the initial solution, $\omega(0)\in\Omega$, and two sets of independent and identically distributed U(0,1) random variables,

i) $\{\xi_i\}$, that generate the neighbors, $\omega \in \eta(\omega(i))$ (hence, allow $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$ to be computed) at each iteration i = 1,2, ...,

ii) $\{\nu_i\}$, that generate values for $R_k(\omega(i), \omega)$, when $\delta(\omega(i), \omega) > 0$, to determine whether the generated neighbor is accepted or rejected (i.e., $R_k(\omega(i), \omega) \geq \delta(\omega(i), \omega)$ or $R_k(\omega(i), \omega) < \delta(\omega(i), \omega)$, respectively).

Note also that $\{\xi_i\}$ and $\{\nu_i\}$ are independent. In addition, assume that when comparing different GHC algorithms, the initial solution is given and fixed across all such algorithms. This means that $(\{\xi_i\},\{\nu_i\})$ completely define the probability of each sequence of I solutions generated by a GHC algorithm. More specifically, let $\Sigma \equiv \Sigma_{\omega(0)}$ denote all sequences of I solutions generated by a GHC algorithm, where each of the I solutions is in Ω, $\Im$ denotes the sigma field of events on Σ, and $P \equiv P_{\xi,\nu}$ denote the probability measure. Therefore, $(\Sigma, \Im, P)$ defines a probability space on the sequences of I solutions generated by a GHC algorithm, where this probability space is characterized by the initial solution, $\omega(0)$, and the set of independent and identically distributed U(0,1) random variables, $(\{\xi_i\},\{\nu_i\})$, that determine the sequence of I solutions generated by a GHC algorithm. For simplicity and ease of notation, $\omega(0)$ and $(\{\xi_i\},\{\nu_i\})$ will be suppressed, unless they are needed to avoid ambiguities.

The iterations of a GHC algorithm can be classified as either micro or macro iterations (Sullivan 1999). A *micro iteration* moves the algorithm from the current solution either to an immediate neighbor or back to itself. A *macro iteration* is a set of micro iterations that moves the algorithm from any element of $G \cup L$ to any element of $G \cup L$ (including itself), passing only through elements of H.

Using these definitions, at macro iteration k (fixed), the micro iterations define a homogeneous discrete time Markov chain, with *micro iteration transition matrix*

$$P_m^k = \begin{bmatrix} P_m^k(G,G) & P_m^k(G,L) & P_m^k(G,H) \\ P_m^k(L,G) & P_m^k(L,L) & P_m^k(L,H) \\ P_m^k(H,G) & P_m^k(H,L) & P_m^k(H,H) \end{bmatrix}$$

where $P_m^k(U,V)$, $U,V \in \{G,L,H\}$ are $|U| \times |V|$ matrices representing the micro iteration transition probabilities from the elements in set U to the elements in set V. Note that if the micro iteration transition Markov chain is aperiodic and irreducible, then at macro iteration k, the macro

$$P_M^k = \begin{bmatrix} P_m^k(G,H)(I - P_m^k(H,H))^{-1} P_m^k(H,G) + P_m^k(G,G) & P_m^k(G,H)(I - P_m^k(G,H))^{-1} P_m^k(H,L) + P_m^k(G,L) \\ P_m^k(L,H)(I - P_m^k(H,H))^{-1} P_m^k(H,G) + P_m^k(L,G) & P_m^k(L,H)(I - P_m^k(H,H))^{-1} P_m^k(H,L) + P_m^k(L,L) \end{bmatrix}$$

iterations define an inhomogeneous Markov chain, with *macro iteration transition matrix.*

Without loss of generality, assume that the GHC algorithm run is initialized at a solution in L (i.e., $\omega(0) \in L$), since local search can be applied from any element in Ω, and the local search property holds for the solution space Ω. This places a restriction on the classes of discrete optimization problems that can be studied, since if a local optima cannot be obtained in polynomial time in the size of the problem instance, then initializing the GHC algorithm run in this way may not be feasible (see Jacobson and Solow 1993). In addition, if local search is applied and the local optima obtained is a G-local optima, then the problem is solved, though this may not be known until further algorithm iterations are executed.

To illustrate the micro iteration and macro iteration concepts, from the GHC algorithm pseudo-code, let the outer loops represent macro iterations, while the inner loops represent micro iterations between the macro iterations. Therefore, as the GHC algorithm is now defined, there will be K macro iterations, with N(k) micro iterations associated with macro iteration k = 1, 2, ..., K (i.e., N(k) micro iterations between macro iteration k-1 and k, where the 0^{th} macro iteration is defined as the initialization of the GHC algorithm run at $\omega(0) \in L$). Note that all the solutions visited during the N(k) micro iterations associated with macro iteration k will be in H, and that N(k), k = 1,2,...,K, will be random variables.

To obtain the necessary and sufficient convergence conditions for a GHC algorithm to converge, define Π_ω^k to be the long run stationary distribution of a GHC algorithm being at solution $\omega \in G \cup L$ at

Using these definitions, at macro iteration k (fixed), the micro iterations define a homogeneous discrete time Markov chain, with *micro iteration transition matrix*

$$P_m^k = \begin{bmatrix} P_m^k(G,G) & P_m^k(G,L) & P_m^k(G,H) \\ P_m^k(L,G) & P_m^k(L,L) & P_m^k(L,H) \\ P_m^k(H,G) & P_m^k(H,L) & P_m^k(H,H) \end{bmatrix}$$

where $P_m^k(U,V)$, $U,V \in \{G,L,H\}$ are $|U| \times |V|$ matrices representing the micro iteration transition probabilities from the elements in set U to the elements in set V. Note that if the micro iteration transition Markov chain is aperiodic and irreducible, then at macro iteration k, the macro

$$P_M^k = \begin{bmatrix} P_m^k(G,H)(I - P_m^k(H,H))^{-1} P_m^k(H,G) + P_m^k(G,G) & P_m^k(G,H)(I - P_m^k(G,H))^{-1} P_m^k(H,L) + P_m^k(G,L) \\ P_m^k(L,H)(I - P_m^k(H,H))^{-1} P_m^k(H,G) + P_m^k(L,G) & P_m^k(L,H)(I - P_m^k(H,H))^{-1} P_m^k(H,L) + P_m^k(L,L) \end{bmatrix}$$

iterations define an inhomogeneous Markov chain, with *macro iteration transition matrix.*

Without loss of generality, assume that the GHC algorithm run is initialized at a solution in L (i.e., $\omega(0) \in L$), since local search can be applied from any element in Ω, and the local search property holds for the solution space Ω. This places a restriction on the classes of discrete optimization problems that can be studied, since if a local optima cannot be obtained in polynomial time in the size of the problem instance, then initializing the GHC algorithm run in this way may not be feasible (see Jacobson and Solow 1993). In addition, if local search is applied and the local optima obtained is a G-local optima, then the problem is solved, though this may not be known until further algorithm iterations are executed.

To illustrate the micro iteration and macro iteration concepts, from the GHC algorithm pseudo-code, let the outer loops represent macro iterations, while the inner loops represent micro iterations between the macro iterations. Therefore, as the GHC algorithm is now defined, there will be K macro iterations, with N(k) micro iterations associated with macro iteration k = 1, 2, ..., K (i.e., N(k) micro iterations between macro iteration k-1 and k, where the 0^{th} macro iteration is defined as the initialization of the GHC algorithm run at $\omega(0) \in L$). Note that all the solutions visited during the N(k) micro iterations associated with macro iteration k will be in H, and that N(k), k = 1,2,...,K, will be random variables.

To obtain the necessary and sufficient convergence conditions for a GHC algorithm to converge, define Π_ω^k to be the long run stationary distribution of a GHC algorithm being at solution $\omega \in G \cup L$ at

macro iteration k. The following lemma provides upper and lower bounds on the sum of the long run stationary distribution of being at a solution in L at macro iteration k.

Lemma 1 (Sullivan and Jacobson 2000a): For a GHC algorithm at macro iteration k (fixed), with macro iteration transition matrix P_M^k,

$$\min_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma) / [\min_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma) + \max_{\sigma \in L} \sum_{\omega \in G} P_M^k(\sigma,\omega)]$$
$$\leq \sum_{\omega \in L} \Pi_\omega^k \leq$$
$$\max_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma) / [\max_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma) + \min_{\sigma \in L} \sum_{\omega \in G} P_M^k(\sigma,\omega)]$$

These bounds are used to provide the necessary and sufficient convergence conditions for a GHC algorithm to converge in probability to G, as given by Theorem 1.

Theorem 1 (Sullivan and Jacobson 2000a): For a GHC algorithm at macro iteration k (fixed), with macro iteration transition matrix P_M^k,

a) (Necessary Condition)
If $\Pi_\omega^k \to 0$ as $k \to +\infty$, for all $\omega \in L$, then
$[\min_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma)] / [\max_{\sigma \in L} \sum_{\omega \in G} P_M^k(\sigma,\omega)] \to 0$ as $k \to +\infty$,

b) (Sufficient Condition)
If $[\max_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma)] / [\max_{\omega \in G} \sum_{\sigma \in L} P_M^k(\omega,\sigma) + \min_{\sigma \in L} \sum_{\omega \in G} P_M^k(\sigma,\omega)] \to 0$
as $k \to +\infty$, then $\Pi_\omega^k \to 0$ as $k \to +\infty$, for all $\omega \in L$.

Theorem 1 uses the macro iteration structure and the macro iteration transition matrix described above to provide conditions that can be used to determine when a GHC algorithm does or does not converge. These conditions are being studied and analyzed to determine convergence rates for different classes of GHC algorithms, as well as to determine how different GHC algorithm formulations can be compared and evaluated for various discrete optimization problems.

3. Performance Measures

An important breakthrough has been the development of the false negative probability for GHC algorithms, and the resulting new necessary convergence condition. To obtain these new results, several important research development milestones had to be attained. All these results are reported in Jacobson and Yucesan (2000a). To describe these developments, a number of additional definitions are needed.

Define A to be a GHC algorithm applied to an instance of a discrete optimization problem, where K represents the *total* number of macro iterations obtained during the algorithm's execution. Assume that

for A, $R_k(\omega(i),\omega) \geq 0$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, for all macro iterations k = 1,2,...,K. At each macro iteration k, define the events on the probability space $(\Sigma, \Im, P)$,

$$B_A(k,G) = B(k,G) \equiv \{\text{A does not visit any element of G after k macro iterations}\} \quad (1)$$

and

$$B_A(G) = B(G) \equiv \{\text{A does not visit any element of G}\}. \quad (2)$$

These two events are distinct in that B(k,G), k = 1,2,...,K, are for algorithm A executed over a finite number of macro iterations k, while B(G) has no such limitation. The complementary events, $B^c(k,G)$ (termed the finite global visit probability) and $B^c(G)$ (termed the global visit probability), are defined as

$$B^c(k,G) \equiv \{\text{A visits at least one element of G after k iterations}\} \quad (3)$$

and

$$B^c(G) \equiv \{\text{A visits at least one element of of G}\}. \quad (4)$$

The definition of B(k,G) in (1) implies that $B(k,G) \supseteq B(k+1,G)$, for all macro iterations k=1,2,...,K, hence {B(k,G)} is a telescoping, non-increasing sequence of events in k. Therefore, since the probability function is a continuous set function, then by the Monotone Convergence Theorem (Ross 1988, p.44),

$$P\{B(K,G)\} \rightarrow P\{B(G)\} \text{ as } K \rightarrow +\infty,$$

where

$$B(G) = \bigcap_{k=1}^{+\infty} B(k,G).$$

After K macro iterations, algorithm A yields K solutions, $\{\omega_1,\omega_2,\ldots,\omega_K\} \subseteq G \cup L$. Define f^K to be the minimum objective function value among these K solutions and ω^K to be the associated solution (i.e., $f^K = f(\omega^K)$ with $\omega^K = \text{argmin}\{f(\omega_k), k = 1,2,\ldots,K\}$). In practice, the best solution to date (i.e., ω^K) is reported. The key issue is whether $\omega^K \in G$. If $\omega^K \in G$, then algorithm A would be terminated at (no later than) macro iteration K. On the other hand, if $\omega^K \notin G$, it would be desirable to determine whether algorithm A will at some future macro iteration find any solution in G. In this case, it would also be desirable to determine the number of additional macro iterations required to visit such a solution. The probability that ω^K is an element of G, $P\{\omega^K \in G\}$, is given by $P\{B^c(K, G)\}$. If this probability is sufficiently close to one, then algorithm A may be terminated. Therefore $P\{B^c(K, G)\}$ provides a quality measure for the solutions obtained after K macro iterations.

To establish the relationship between the convergence of a GHC algorithm and B(G), the following definition of convergence in probability is formally stated.

DEFINITION 1: A GHC algorithm A *converges in probability to G* if $P\{C(K,G)\} \rightarrow 1$ as $K \rightarrow +\infty$, where $C_A(K,G) = C(K,G) \equiv \{A \text{ is at an element of G at macro iteration K}\} = \{\omega_K \in G\}$.

Therefore, for a given (fixed) initial solution $\omega(0)$, if algorithm A converges in probability to G (as $K \rightarrow +\infty$), then $P\{B^c(G)\} = 1$. Equivalently, if $P\{B^c(G)\} < 1$, then algorithm A does not converge in probability to G (i.e., there exists some $\varepsilon > 0$ and a macro iteration K_0 such that $P\{C(K,G)\} \leq 1 - \varepsilon$ for all $K \geq K_0$). The convergence behavior of GHC algorithms is further investigated in Section 4.

In light of these observations, the *false negative problem* asks whether a GHC algorithm A will eventually visit an element of G, given that the algorithm, after executing a finite number of macro iterations, has yet to visit an element of G. This question can be quantified by considering the *false negative probability* at macro iteration K, $P\{B^c(G) \mid B(K,G)\}$.

The false negative probability at macro iteration K provides a measure for the effectiveness of a GHC algorithm A, namely the ability of algorithm A to visit an element of G beyond macro iteration K. In particular, if $P\{B^c(G)\}$ is small, then one can use the false negative probability to assess whether a GHC algorithm will eventually visit an element of G; if the false negative probability at macro iteration K is sufficiently close to zero, then the algorithm may be terminated.

The false negative probability at macro iteration K can be related to the Type II error in a hypothesis testing framework. To see this, consider the following hypotheses in the search for a globally optimal solution using a GHC algorithm:

H_0: There does not exist a solution $\omega \in \Omega$ with objective function value at or below θ
(i.e., $f(\omega) > \theta$ for all $\omega \in \Omega$),

H_A: There exists a solution $\omega \in \Omega$ with objective function value at or below θ
(i.e., $f(\omega) \leq \theta$ for some $\omega \in \Omega$),

where $\theta \in R$. If $\theta = f(\omega^*)$, $\omega^* \in G$, then by viewing the GHC algorithm A as a sampling procedure over the solution space, Ω, a Type I error cannot occur, since if algorithm A finds a solution with objective function value at or below θ, then H_0 cannot be falsely rejected. However, a Type II error can occur, with probability $\beta_K = P\{B(K,G)\}$. Using the law of total probability and Bayes Theorem, the probability of a false negative can be expressed as $P\{B^c(G) \mid B(K,G)\} = 1 - P\{B(G)\} / \beta_K$. This relationship suggests three possible cases:

Case a: $P\{B^c(G)\} = 0$. This is the case where algorithm A is guaranteed to never visit any element of G, irrespective of the number of macro iterations. For this situation, the false negative probability is zero for all macro iterations $k = 1,2,\ldots$, since $\beta_k = 1$ for all $k = 1,2,\ldots$.

Case b: $P\{B^c(G)\} = 1$. This is the case where algorithm A is guaranteed to visit an element of G, provided that a sufficiently large number of macro iterations are executed. For this situation, the false negative probability is one for all macro iterations k=1,2,....

Case c: $0 < P\{B^c(G)\} < 1$. This is the case where algorithm A may visit an element of G, though this cannot be guaranteed. For this situation, the false negative probability is between zero and one for all macro iterations k = 1,2,..., since $P\{B(G)\} < \beta_k$ for all macro iterations k=1,2,.... Moreover, the false negative probability is inversely related to the power of the test, $1 - \beta_k$ (i.e., the GHC algorithm's ability to visit an element of Ω with objective function value at or below θ).

Each of these three cases is encountered in practice. Case a occurs, for example, if the algorithm is a local search procedure and with probability one gets trapped in a L-local optimum. Case b occurs, for example, if the algorithm is Monte Carlo search or simulated annealing with a convergent cooling schedule. Case c occurs, for example, if the algorithm is simulated annealing with a cooling schedule that does not guarantee convergence.

The false negative probability can be used to derive necessary convergence conditions for GHC algorithms. The false negative probability is also used to measure the effectiveness of non-convergent (in probability) GHC algorithms. Recall that $P\{B(0, G)\} = 1$ (i.e., all GHC algorithm runs are initialized at an element of L, hence $\omega(0) \in L$). Furthermore, unless otherwise stated, assume that $P\{B^c(k, G)\} < 1$ for all macro iterations k = 1,2,...,K.

For macro iteration k, define the event

$$R_A(k,G) = R(k,G) \equiv \{\text{A visits any element of G after k macro iterations, given that A has not visited any element of G after k-1 macro iterations}\}, \quad (5)$$

where

$$r(k,G) \equiv P\{R(k,G)\} = P\{B^c(k,G) \mid B(k\text{-}1,G)\} = P\{C(k,G) \mid B(k\text{-}1,G)\}. \quad (6)$$

This probability can be used to quantify the false negative probability. Lemma 2 expresses the relationship between (6) and (1).

LEMMA 2: (i) $P\{B(K,G)\} = \prod_{k=1}^{K} [1 - r(k,G)]$ for all macro iterations K.

(ii) $P\{B(G)\} = \prod_{k=1}^{+\infty} [1 - r(k,G)]$.

Proof: See Jacobson and Yucesan (2000a).

Lemma 3 provides an expression for the *finite false negative probability,* $P\{B^c(J,G) \mid B(K,G)\}$, $J > K$.

LEMMA 3: For all macro iterations J,K, with J>K, $P\{B^c(J,G) \mid B(K,G)\} = 1 - \prod_{k=K+1}^{J} [1 - r(k,G)]$.

Proof: See Jacobson and Yucesan (2000a).

Theorem 2 provides a closed-form expression for the false negative probability.

THEOREM 2: For all macro iterations K, $P\{B^c(G) \mid B(K,G)\} = 1 - \prod_{k=K+1}^{+\infty} [1-r(k,G)]$.

Proof: See Jacobson and Yucesan (2000a).

Corollary 1 shows that the false negative probability is non-increasing in K.

COROLLARY 1: For all macro iterations K, $P\{B^c(G) \mid B(K,G)\} \geq P\{B^c(G) \mid B(K+1,G)\}$.

Proof: See Jacobson and Yucesan (2000a).

Corollary 1 establishes that the probability of visiting any element of G never increases with each successive macro iteration that has not visited any element of G. This result can be used as a guideline for execution termination. For example, consider two GHC algorithms A_1 and A_2 that are not guaranteed to converge in probability to G. If after K macro iterations, suppose that algorithm A_1 has a smaller false negative probability than algorithm A_2, though the best solution obtained thus far by algorithm A_1 has lower objective function value than the best solution obtained thus far by algorithm A_2. Then, if one of the algorithms must be terminated, one may choose to terminate algorithm A_2 since its performance over K macro iterations has been inferior to that of algorithm A_1. However, since the false negative probability is a *measure of future performance*, then algorithm A_1 should be terminated, because algorithm A_1 is less likely than algorithm A_2 to visit any element of G in subsequent macro iterations, given that neither algorithm has visited any element of G after K macro iterations. Note that the number of micro iterations between each macro iteration must also be taken into account when deciding upon which algorithm to terminate. Initial results on the expected value for this measure are reported in Jacobson and Yucesan (2000a).

Theorem 3 expresses the probability that a GHC algorithm does not visit any element of G (after K macro iterations), given that it *eventually* does visit an element of G in J > K macro iterations (where J could be infinite). This result provides a marginal value measure for executing additional iterations.

THEOREM 3: For all macro iterations J,K, with J > K and $P\{B^c(J,G)\} > 0$,

$$P\{B(K,G) \mid B^c(J,G)\} = \{\prod_{k=1}^{K} [1-r(k,G)] - \prod_{j=1}^{J} [1-r(j,G)]\} / \{1-\prod_{j=1}^{J} [1-r(j,G)]\}.$$

Proof: See Jacobson and Yucesan (2000a).

COROLLARY 2: For all macro iterations K and $P\{B^c(G)\} > 0$,

$$P\{B(K,G) \mid B^c(G)\} = \{\prod_{k=1}^{K} [1-r(k,G)] - \prod_{j=1}^{+\infty} [1-r(j,G)]\} / \{1-\prod_{j=1}^{+\infty} [1-r(j,G)]\}.$$

Proof: See Jacobson and Yucesan (2000a).

Note that the probability in Corollary 2 is non-increasing in K. Therefore, for all $\varepsilon > 0$ (close to 0), there exists a macro iteration K_0, such that, for all macro iterations $K \geq K_0$, $P\{B(K,G) \mid B^c(G)\} \leq \varepsilon$. This can be used to terminate a GHC algorithm once the probability in Corollary 2 is sufficiently small, since the marginal value of each additional macro iteration is negligible. Moreover, since the probability that an element of G has been visited over the first K_0 macro iterations, given that an element of G is eventually visited, is at least $1-\varepsilon$, then it is unlikely that a premature termination has occurred.

Theorem 4 provides upper and lower bounds for the finite false negative probability.

THEOREM 4: For all macro iterations J,K, with J > K, and r(k,G) < 1 for all k = 1,2,...,J,

$$1 - \exp\{-\sum_{k=K+1}^{J} r(k,G)\} \leq P\{B^c(J,G) \mid B(K,G)\} \leq 1 - \exp\{-\sum_{k=K+1}^{J} [r(k,G)] / [1 - r(k,G)]\}.$$

Proof: See Jacobson and Yucesan (2000a).

Corollary 3 provides upper and lower bounds for the false negative probability.

COROLLARY 3: For all macro iterations K,

$$1 - \exp\{-\sum_{k=K+1}^{+\infty} r(k,G)\} \leq P\{B^c(G) \mid B(K,G)\} \leq 1 - \exp\{-\sum_{k=K+1}^{+\infty} [r(k,G)] / [1 - r(k,G)]\}.$$

Proof: See Jacobson and Yucesan (2000a).

Proposition 1 establishes the relationship between convergence in probability to G and the false negative probability.

PROPOSITION 1: If a GHC algorithm A converges in probability to G, then GHC algorithm A *visits G in probability* (i.e., $P\{B^c(G) \mid B(K,G)\} = 1$ for all macro iterations K = 1,2,...).

Proof: See Jacobson and Yucesan (2000a).

From Proposition 1, if a GHC algorithm converges in probability to G, then the GHC algorithm A visits G in probability. Proposition 2 provides necessary and sufficient conditions for the false negative probability to be one for all macro iterations.

PROPOSITION 2: A GHC algorithm A visits G in probability if and only if

$$\sum_{k=K+1}^{+\infty} r(k,G) = +\infty \text{ for all macro iterations } K = 1,2,....$$

Proof: See Jacobson and Yucesan (2000a).

Proposition 3 establishes the relationship between the false negative probability and $P\{B^c(G)\}$.

PROPOSITION 3: A GHC algorithm A visits G in probability if and only if $P\{B^c(G)\} = 1$.

Proof: See Jacobson and Yucesan (2000a).

Theorem 5 summarizes the relationship between the global visit probability, the false negative probabilities, r(K,G), and convergence in probability to G.

THEOREM 5: For a GHC algorithm A, consider the expressions

(D1) $P\{C(K,G)\} \rightarrow 1$ as $K \rightarrow +\infty$ (converges in probability to G).

(D2) $P\{B^c(G) \mid B(K,G)\} = 1$ for all macro iterations K (visits G in probability).

(D3) $P\{B^c(G)\} = 1$ (visits G in probability).

(D4) $\sum_{k=K+1}^{+\infty} r(k,G) = +\infty$ for all macro iterations K.

Then, (D1) $\Rightarrow$ (D2) $\Leftrightarrow$ (D3) $\Leftrightarrow$ (D4).

Proof: Follows from Propositions 1, 2, and 3.

Theorem 5 provides three necessary conditions for the convergence of a GHC algorithm. The only restriction on how the GHC algorithm traverses the solution space is that $P\{B(K,G)\} > 0$ for all macro iterations K = 1,2,.... This restriction means that the GHC algorithm is never guaranteed to visit any element of G for K finite (i.e., $P\{B^c(K,G)\} < 1$ for all macro iterations K = 1,2,...).

From Lemma 1, if $P\{B(G)\} = \prod_{k=1}^{+\infty} [1-r(k,G)] > 0$, then, from Theorem 5, the GHC algorithm does not converge in probability to G. Theorem 6 provides an upper bound on the probability associated with the GHC algorithm converging in probability to G.

THEOREM 6: For a GHC algorithm A, $P\{C(K,G)\} \leq 1 - \psi$ for all macro iterations K = 1,2,..., where $\psi = \prod_{k=1}^{+\infty} [1-r(k,G)]$. Therefore, if $0 < \psi \leq 1$, then GHC algorithm A does not converge in probability to G.

Proof: See Jacobson and Yucesan (2000a).

Theorem 7 shows that for a GHC algorithm with $\sum_{k=1}^{+\infty} r(k,G) < +\infty$, then with probability one, only a finite number of the events R(k,G), k = 1,2,..., occur simultaneously, or equivalently, with probability zero, the R(k,G), k = 1,2,..., occur infinitely often.

THEOREM 7: Suppose that for a GHC algorithm A, $\sum_{k=1}^{+\infty} r(k,G) < +\infty$ (hence, the algorithm does not converge in probability to G). Then R(k,G), k = 1,2,..., occur finitely often with probability one.

Proof: See Jacobson and Yucesan (2000a).

Theorem 7 shows that if $\sum_{k=1}^{+\infty} r(k,G) < +\infty$, hence the GHC algorithm does not converge in probability to G, then for all $\varepsilon > 0$, there exists a $K(\varepsilon) \in Z^+$ such that $P\{\bigcup_{k=K(\varepsilon)}^{+\infty} R(k,G)\} \leq \sum_{k=K(\varepsilon)}^{+\infty} r(k,G) \leq \varepsilon$. This means that for some nonconvergent (in probability) GHC algorithms, the probability that it will visit an element of G for the first time (at or beyond macro iteration K) can be made arbitrarily small if the macro iteration K is set sufficiently large. This result can be used to define a stopping condition for a GHC algorithm run. For example, if an improved solution has not been observed for some prespecified number of macro iterations, it may be feasible to terminate the GHC algorithm run.

Theorem 8 establishes a similar result as Theorem 6 using a convergence condition on the 1−r(k,G), k = 1,2,....

THEOREM 8: Suppose that for a GHC algorithm A, $\sum_{k=1}^{+\infty} (1-r(k,G)) < +\infty$. Then R(k,G), k = 1,2,... , occur almost always with probability one.

Proof: See Jacobson and Yucesan (2000a).

Theorem 8 shows that if $\sum_{k=1}^{+\infty} (1-r(k,G)) < +\infty$, then for all $\varepsilon > 0$, there exists a $K(\varepsilon) \in Z^+$ such that $P\{R(K,G)\} \geq P\{\bigcap_{k=K((\varepsilon)}^{+\infty} R(k,G)\} \geq 1 - \sum_{k=K(\varepsilon)}^{+\infty} (1-r(k,G)) \geq 1 - \varepsilon$ for all $K \geq K(\varepsilon)$. This means that the probability that a GHC algorithm (with $\sum_{k=1}^{+\infty} (1-r(k,G)) < +\infty$) will visit an element of G for the first time (at or beyond macro iteration K) can be made arbitrarily close to one if the macro iteration K is set sufficiently large.

These results can be used to assess the performance of various GHC algorithms. In particular, the performance of three GHC algorithms, Monte Carlo search, random-restart local search, and threshold accepting, can be evaluated. For Monte Carlo Search, Theorem 5 implies that the false negative probability is one (for all macro iterations) for Monte Carlo search. To see this, Monte Carlo search can be described as a GHC algorithm by setting $\eta(\omega) = \Omega$ for all $\omega \in \Omega$, and $R_k = +\infty$ for all macro iterations

k=1,2,.... If $p(G) \equiv \gamma / (\gamma+\lambda)$, then $r(k,G) = p(G)$. Therefore, $P\{B(k,G)\} = [1-p(G)]^k$. From Lemma 2, for macro iterations j and k, with $j > k$,

$$P\{B^c(j,G) \mid B(k,G)\} = 1 - [1-p(G)]^{j-k},$$

hence the finite false negative probability approaches one as j approaches infinity. Moreover, from Theorem 5, $\sum_{j=k+1}^{+\infty} r(j,G) = \sum_{j=k+1}^{+\infty} p(G) = +\infty$ for all macro iterations k. This means that Monte Carlo search visits G in probability as k approaches infinity. However, $P\{C(k,G)\} = p(G)$ for all macro iterations k, hence, Monte Carlo search does not converge in probability to G. More specifically, from Theorem 5, (D2), (D3), and (D4) all hold, but (D1) is not satisfied.

Random-restart local search combines Monte Carlo search and local search, by randomly selecting a new initial solution every time a local search algorithm terminates at a local optimum. The analysis for Monte Carlo search also shows that the false negative probability is one (for all macro iterations) for random-restart local search, by redefining p(G) to be the probability that a randomly generated initial element of Ω will terminate at an element of G. Moreover, random-restart local search will not converge in probability to G (i.e., from Theorem 5, (D2), (D3), and (D4) all hold, but (D1) is not satisfied).

Threshold accepting is a particular GHC algorithm with $R_k(\omega(i),\omega) = T_k$, $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, for macro iteration k, where T_k approaches zero as k approaches infinity. Therefore, there exists an $\varepsilon > 0$ sufficiently small and a macro iteration k_0 such that $|T_k| < \varepsilon$ and $P\{R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)\} = 0$ for all $\omega(i)\in L$, $\omega\in\eta(\omega(i))$, and all $k \geq k_0$, hence (D4) in Theorem 5 does not hold. This implies that this particular form of threshold accepting does not converge in probability to G. However, if T_k is set such that it does not approach zero, hence $r(k,G) \geq \delta$ for some $\delta > 0$ and for all macro iterations k, then (D4) in Theorem 5 may hold and the probability of a false negative is one at all macro iterations k. However, setting T_k in this way may not be feasible in practice, since it requires full knowledge of the solution space (with respect to the depth of all the L-local and G-local optima; see Hajek 1988).

These results introduce the false negative probability as a performance measure that reflects how effectively a GHC algorithm has performed to date as well as how effectively a GHC algorithm can be expected to perform in the future. This analysis also presents expressions and bounds for the false negative probability within the GHC algorithm framework. Work is in progress to determine how the false negative probability can be used to develop guidelines to design effective run strategies for GHC algorithms that do not converge in probability to G. These guidelines can be used, for example, to determine termination conditions once the marginal value of additional iterations is deemed negligible.

Work is also in progress to extend the application of the false negative probability as well as to identify other new performance measures for GHC algorithms.

4. Benchmark Results for Generalized Hill Climbing Algorithms

The false negative probability measure for GHC algorithms can be used to benchmark the performance of GHC algorithms. This resulted in new necessary and sufficient convergence conditions, as well as a way to use new performance measures to establish benchmark performance criteria. These results are reported in Jacobson and Yucesan (2000b). To describe these results, several additional definitions are needed.

Proposition 4 establishes the well known relationship between convergence in probability to G, almost sure convergence to G, and visits G infinitely often.

PROPOSITION 4: Let A be a GHC algorithm.

i) If GHC algorithm A converges almost surely to G (as the number of macro iterations approach infinity), then algorithm A converges in probability to G.

ii) If GHC algorithm A converges in probability to G, then algorithm A visits G infinitely often.

Proof: See Jacobson and Yucesan (2000b).

Proposition 5 provides a condition on the hill climbing random variables such that i) and ii) in Proposition 4 become the same.

PROPOSITION 5: Let A be a GHC algorithm. Suppose that there exists a macro iteration K_0 such that $R_k(\omega,\omega') = 0$ with probability one for all $\omega \in \Omega$, $\omega' \in \eta(\omega)$, $k \geq K_0$. Then a GHC algorithm A converges almost surely to G (as the number of macro iterations approaches infinity) if and only if it visits G infinitely often.

Proof: See Jacobson and Yucesan (2000b).

From Proposition 5, if for GHC algorithm A there exists a macro iteration K_0 such that $R_k(\omega,\omega') = 0$ with probability one for all $\omega \in \Omega$, $\omega' \in \eta(\omega(i))$, $k \geq K_0$, then convergence almost surely to G and convergence in probability to G are identical. Therefore, if a GHC algorithm is designed to become pure local search after some finite number of iterations, then convergence in probability to G reduces to almost sure convergence to G. Moreover, from (4), for a given (fixed) initial solution $\omega(0)$, if algorithm A converges in probability to G (as $K \rightarrow +\infty$), then $P\{B^c(G)\} = 1$. Equivalently, if $P\{B^c(G)\} < 1$, then algorithm A does not converge in probability to G (i.e., there exists some $\varepsilon > 0$ and a macro iteration K_0 such that $P\{C(K,G)\} \leq 1-\varepsilon$ for all $K \geq K_0$).

The finite global visit probability, $P\{B^c(k,G)\}$, can be used to derive necessary and sufficient convergence conditions for GHC algorithms. Recall that $P\{B(0,G)\} = 1$ (i.e., all GHC algorithm runs are initialized at an element of L, hence $\omega(0) \in L$). Furthermore, unless otherwise stated, assume that $P\{B^c(k,G)\} < 1$ for all macro iterations $k = 1,2,\ldots,K$.

The *one-step macro iteration transition probability*, $r(k,G)$, defined in (6) is used to obtain the necessary and sufficient convergence conditions. First, Lemma 4 expresses the relationship between (6) and (1).

LEMMA 4 (i) $P\{B(K,G)\} = \prod_{k=1}^{K} [1 - r(k,G)]$ for all macro iterations K.

(iii) $P\{B(G)\} = \prod_{k=1}^{+\infty} [1 - r(k,G)]$.

Proof: See Jacobson and Yucesan (2000b).

Theorem 9 provides necessary and sufficient conditions for a GHC algorithm to converge in probability to G.

THEOREM 9: Let A be a GHC algorithm. Then $P\{C(K,G)\} \rightarrow 1$ as $K \rightarrow +\infty$ (i.e., converges in probability to G) if and only if

i) $\sum_{K=1}^{+\infty} r(K,G) = +\infty$,

ii) $P\{C^c(K,G) \mid B^c(K-1,G)\} \rightarrow 0$ as $K \rightarrow +\infty$.

Proof: See Jacobson and Yucesan (2000b).

Condition i) requires that $r(K,G)$ not converge to zero too quickly, as K approaches plus infinity. This means that the conditional probability that GHC algorithm A visits an element of G for the first time at macro iteration K approaches zero sufficiently slowly such that the infinite summation diverges. Condition ii) requires that the conditional probability that GHC algorithm A visits an element of G for either the second, or the third, or up to the K^{th} time, including the K^{th} macro iteration, approaches one as the number of macro iterations K approaches plus infinity.

If a GHC algorithm does not converge in probability to G, then conditions i) and ii) in Theorem 9 suggest three possible cases:

Case 1) $P\{C^c(K,G) \mid B^c(K-1,G)\} \rightarrow 0$ as $K \rightarrow +\infty$, but $\sum_{K=1}^{+\infty} r(K,G) < +\infty$.

In this case $\sum_{K=1}^{+\infty} r(K,G) < +\infty$ implies that $P\{B^c(G)\} < 1$.

Therefore, $P\{C(K,G)\} \rightarrow P\{B^c(G)\} = 1 - \prod_{k=1}^{+\infty} [1-r(k,G)] < 1$ as $K \rightarrow +\infty$.

Case 2) $\sum_{K=1}^{+\infty} r(K,G) = +\infty$, but $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\} \not\rightarrow 0$ as $K \rightarrow +\infty$.

If $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\} \geq \varepsilon > 0$ for all $K \geq K_0$ for some $K_0 \in Z^+$, then there exists some $K_1 \in Z^+$, $K_1 \geq K_0$, such that $P\{C(K,G)\} \leq 1\text{-}\varepsilon$ for all $K \geq K_1$,

Case 3) $\sum_{K=1}^{+\infty} r(K,G) < +\infty$ and $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\} \not\rightarrow 0$ as $K \rightarrow +\infty$.

If $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\} \geq \varepsilon > 0$ for all $K \geq K_0$ for some $K_0 \in Z^+$, then there exists some $K_1 \in Z^+$, $K_1 \geq K_0$, such that then $P\{C(K,G)\} \leq 1 - \varepsilon P\{B^c(G)\} + \varepsilon'(K_1)$ for all $K \geq K_1$, where $P\{B^c(G)\}>0$ and $\varepsilon'(K_1)$ can be made arbitrarily small for K_1 sufficiently large.

For Case 1), $P\{C(K,G)\}$ converges to the global visit probability, as $K \rightarrow +\infty$. For Case 2), $P\{C(K,G)\}$ is bounded above by one minus a strictly positive lower bound for $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\}$. For Case 3), $P\{C(K,G)\}$ is bounded above by the global visit probability times one minus a strictly positive lower bound for $P\{C^c(K,G) \mid B^c(K\text{-}1,G)\}$ (plus a value that can be made arbitrarily close to zero). These three cases illustrate the relationship between the global visit probability and the convergence in probability to an element of G of a GHC algorithm.

The conditions in Theorem 9 can be related to the convergence conditions for simulated annealing presented in Hajek (1988). In particular, Hajek (1988) shows that simulated annealing converges in probability to the set of global optima (i.e., G-local optima) if and only if $\sum_{k=1}^{+\infty} e^{-(d^*/t(k))} = +\infty$, where t(k) is a nonincreasing cooling schedule at iteration k (that approaches zero as $k \rightarrow +\infty$), and d* is the maximum depth of all elements in L (i.e., the maximum gap in objective function value between an element of L and the solution in H that can reach another element of $L \cup G$ via local search, where the maximum is taken over all elements of L). This result assumes that the depth of all elements in G is infinity, hence once a G-local optima is reached, a simulated annealing cannot escape (with probability one) from such an element. Therefore, if GHC algorithm A is simulated annealing, then under this assumption, ii) in Theorem 9 is always satisfied. Moreover, since the solution space is reachable, then at each macro iteration K sufficiently large, there is a positive probability that the algorithm will need to escape from each element of L and move to an element of G. In particular, at each macro iteration K sufficiently large, the conditional probability r(K,G | A) has a component that includes the probability of escaping from the deepest L-local optimum. Therefore, using the law of total probability,

$$r(K,G \mid A) = \Sigma_{\omega \in L}\, r(K,G \mid A, \omega \in L \text{ is visited at macro iteration } K-1\}$$
$$P\{\omega \in L \text{ is visited at macro iteration } K-1\} \qquad (7)$$

Therefore, there exists a lower bound for $r(K,G \mid A)$ that is a linear function of P{moving from the deepest element of L to an element of G} = P{Accepting hill climbing moves out of the deepest element of L to an element of G} = $O(^{-(d^*/t(K))})$, since the hill climbing random variable at macro iteration K is exponential with mean $1/t(K)$. Therefore, a sufficient condition for i) in Theorem 9 is $\sum_{K=1}^{+\infty} e^{-(d^*/t(K))} = +\infty$.

To establish that this infinite summation is a necessary condition for i) in Theorem 9, using the same analysis as described above, at each macro iteration K sufficiently large, the conditional probability $r(K,G \mid A)$ has a component (see (7)) that includes the probability of escaping from each element of L. This means that for each element of L, there exists a component for $r(K,G \mid A)$ that is $O(e^{-(d(\omega)/t(K))})$, $\omega \in L$, where $d(\omega)$ is defined as the depth of the L-local optimum ω. Therefore, if i) in Theorem 9 holds, then the infinite summation over all of these components must be unbounded, or else with positive probability, the algorithm visits an element of L and be unable to move from it to another element of $L \cup G$. This means that $\sum_{K=1}^{+\infty} r(K,G \mid A, \omega \in L \text{ is visited at macro iteration } K-1\} = +\infty$ for all $\omega \in L$, which establishes that $\sum_{K=1}^{+\infty} e^{-(d^*/t(K))} = +\infty$ is a necessary conditions for i) in Theorem 9.

The finite global visit probability can also be used to compare different GHC algorithms. In particular, *random restart local search* (RR) is used as a benchmark algorithm to compare the performance of various GHC algorithms. Let LS denote a single macro iteration (restart) of random restart local search. Random restart local search involves randomly selecting an initial solution (i.e., uniformly generated over the entire solution space, hence for all $\omega \in \Omega$, P{Algorithm LS is initialized at $\omega \in \Omega\} = 1/|\Omega|$), and applying local search until a local optimum is found. This process is repeated until K local optima are obtained. The best of these K solutions is then reported as the final solution. Therefore, each random restart corresponds to a single macro iteration. Using the GHC algorithm framework described in Section 2.1, the hill climbing (random) variables $R_k(\omega(i),\omega) = 0$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, k = 1,2,...,K, with N(k) representing the number of micro iterations needed to reach the k^{th} element of $L \cup G$. In addition, after this element of $L \cup G$ is found, a new element of Ω is randomly (uniformly) generated to begin the next (inner loop) set of micro iterations.

To show how random restart local search can be used to compare different GHC algorithms, at each macro iteration (i.e., at each random restart), define the conditional probability

$$p(\omega) \equiv P\{\text{Algorithm LS terminates in G} \mid \text{Algorithm LS is initialized at } \omega \in \Omega\}. \qquad (8)$$

Using the law of total probability, define

$$\begin{aligned} P\{\text{G-stop}\} &\equiv P\{\text{Algorithm LS terminates in G}\} \\ &= \sum_{\omega\in\Omega} p(\omega) * P\{\text{Algorithm LS is initialized at } \omega\} \\ &= [\ |G| + \sum_{\omega\in H} p(\omega)\ ] / \ |\Omega|. \end{aligned} \tag{9}$$

Similarly,

$$\begin{aligned} P\{\text{L-stop}\} &\equiv P\{\text{Algorithm LS terminates in L}\} \\ &= 1 - P\{\text{G-stop}\} \\ &= [\ |L| + \sum_{\omega\in H} (1 - p(\omega))\ ] / \ |\Omega|. \end{aligned} \tag{10}$$

Note that P{L-stop} is defined by the neighborhood function and the neighborhood probability mass function (which is defined to be uniform). Therefore, if η_1 and η_2 are two neighborhood functions defined on a solution space Ω, where $\eta_1(\omega) \subseteq \eta_2(\omega)$ for all $\omega\in\Omega$, then $L(\eta_1) \supseteq L(\eta_2)$ and P{L-stop} for η_1 is greater than or equal to P{{L-stop} for η_2. In general, enriching the neighborhood function such that L-local optima are eliminated tends to decrease P{L-stop}.

Without loss of generality, assume that the neighborhood function is defined such that P{L-stop}>0. If this is not the case, then P{G-stop}=1, hence local search will always find a G-local optimum with every restart (i.e., $P\{(B_{RR}(K,G))^c\}=1$ for all K=1,2,...). Under this assumption, Theorem 10 provides a convergence comparison between random restart local search and a GHC algorithm that does not visit G in probability.

THEOREM 10: Let A be a GHC algorithm that does not visit G in probability. Let RR be a random restart local search algorithm. If the neighborhood function and the neighborhood probability generation function are defined on the solution space such that $0<P\{\text{L-stop}\}<1$, then there exists a macro iteration K_0 (which represents the number of restarts for the random restart local search algorithm) such that for all $K\geq K_0$,

$$P\{(B_{RR}(K,G))^c\} \geq P\{(B_A(K,G))^c\}.$$

Moreover, $K_0 \leq \ln[\alpha]/\ln[P\{\text{L-stop}\}]$, where $\prod_{k=1}^{+\infty} (1 - r(k,G)) \geq \alpha > 0$.

Proof: See Jacobson and Yucesan (2000b).

Theorem 10 shows that if a GHC algorithm does not visit G in probability, then there exists a macro iteration beyond which random restart local search yields better results, as measured by the finite global visit probabilities at macro iteration K, $P\{(B_A(K,G))^c\}$ and $P\{(B_{RR}(K,G))^c\}$ (i.e., the probabilities that the

best solution visited to date (i.e., after K macro iterations) is in G, for algorithms A and RR, respectively).

Theorem 11 provides sufficient conditions on the rate at which $P\{B_A(K,G)\}$ converges to zero such that the performance of random restart local search and a GHC algorithm can be compared. To described these conditions, define the non-negative value $\varphi_0 = -\ln(P\{\text{L-stop}\})$.

THEOREM 11: Let A be a GHC algorithm that visit G in probability. Let RR be a random restart local search algorithm. If the neighborhood function and the neighborhood probability generation function are defined on the solution space such that $0<P\{\text{L-stop}\}<1$, then

i) if $P\{B_A(K,G)\} = O(e^{-\varphi K})$ for K large and $\varphi \geq \varphi_0$, then there exists a macro iteration K_0 such that for all $K \geq K_0$, $P\{(B_{RR}(K,G))^c\} \leq P\{(B_A(K,G))^c\}$,

ii) if $P\{B_A(K,G)\} = O(e^{-\varphi K})$ for K large and $\varphi < \varphi_0$, then there exists a macro iteration K_0 such that for all $K \geq K_0$, $P\{(B_{RR}(K,G))^c\} \geq P\{(B_A(K,G))^c\}$,

iii) if $P\{B_A(K,G)\} = o(e^{-\varphi K})$ for K large and $\varphi \geq \varphi_0$, then there exists a macro iteration K_0 such that for all $K \geq K_0$, $P\{(B_{RR}(K,G))^c\} \leq P\{(B_A(K,G))^c\}$,

iv) if $1/P\{B_A(K,G)\} = o(e^{\varphi K})$ for K large and $\varphi \leq \varphi_0$, then there exists a macro iteration K_0 such that for all $K \geq K_0$, $P\{(B_{RR}(K,G))^c\} \geq P\{(B_A(K,G))^c\}$,

where $P\{B_A(K,G)\} = O(e^{-\varphi K})$ for K large means that $P\{B_A(K,G)\}\ e^{\varphi K} \rightarrow$ a constant as $K \rightarrow +\infty$, $P\{B_A(K,G)\} = o(e^{-\varphi K})$ for K large means that $P\{B_A(K,G)\}\ e^{\varphi K} \rightarrow 0$ as $K \rightarrow +\infty$, and $1/P\{B_A(K,G)\} = o(e^{\varphi K})$ for K large means that $1/[P\{B_A(K,G)\}\ e^{\varphi K}] \rightarrow 0$ as $K \rightarrow +\infty$.

Proof See Jacobson and Yucesan (2000b).

Theorem 11 compares the performance of random restart local search and a GHC algorithm that visit G in probability for four different cases. The remaining two cases (i.e., $P\{B_A(K,G)\} = o(e^{-\varphi K})$ for K large with $\varphi < \varphi_0$ and $1/P\{B_A(K,G)\} = o(e^{\varphi K})$ for K large with $\varphi > \varphi_0$) are inconclusive, hence the performance of each of the algorithms becomes problem specific, where general results cannot be obtained using the approach presented here. Note that an identical analysis can be used to show that Monte Carlo search yields the same conclusion obtained in both Theorems 10 and 11, with the new definitions $P\{\text{G-stop}\} \equiv |G| / |\Omega|$ and $P\{\text{L-stop}\} \equiv 1 - P\{\text{G-stop}\}$ (i.e., P{L-stop} is no longer defined as in (10)). However, P{L-stop} for Monte Carlo search will be greater than or equal to P{L-stop} for random restart local search. This means that there exists GHC algorithms for which ii) and iv) in Theorem 11 are satisfied for random restart local search, but are not satisfied for Monte Carlo search, while there are no GHC

algorithms that converge in probability for which the reverse is true. Therefore, random restart local search dominates Monte Carlo search.

The results in Theorem 11 suggest that the value for P{L-stop} for random restart local search determines its relative performance with GHC algorithms that visit G in probability. In particular, if P{L-stop}=1, then ii) and iv) can never be satisfied. This means that random restart local search cannot be proven to dominate such GHC algorithms using Theorem 11. In general, $1-\delta \leq P\{\text{L-stop}\} < 1$ for some $\delta > 0$ close to zero. Therefore, $\varphi_0 = -\ln(P\{\text{L-stop}\}) \geq -\ln(1-\delta) \geq \delta$. This means that the larger the value for P{L-stop} (i.e., the closer to one), the larger the number of restarts needed for random restart local search to dominate a GHC algorithm that visit G in probability. For practical purposes, this suggests that for solution spaces (and associated neighborhood functions) with many local optima, it may be more effective to use a GHC algorithm. Therefore, the design and structure of the neighborhood function (hence the number of and distribution of L-local optima in the solution space) is a key factor in determining whether random restart local search is preferable over a GHC algorithm.

The results in Theorems 10 and 11 do not take into account the number of (micro) iterations between each macro iteration. For random restart local search, this represents the number of iterations needed to reach a local optimum from each randomly generated initial solution, while for a GHC algorithm, this represents the number of iterations between visits to local optima. Note that as a GHC algorithm progresses, and the hill climbing random variables approach the value zero with probability one, the number of iterations between the macro iterations may be very small, as the algorithm gets stuck in the same local optimum with increasing probability. If this local optimum is a L-local optimum, then $P\{(B_A(K,G))^c\}$ will remain constant (or change very slowly) for all future macro iterations. Fox (1993, 1995) notes this point for simulated annealing, and suggests alternate ways to improve the performance of simulated annealing that overcomes this situation.

The results in Theorem 11 have important implications for practitioners using simulated annealing. Using the necessary and sufficient convergence condition for simulated annealing in Hajek (1988), r(K,G) can be bounded above and below by functions that are $O(e^{-(d^*/t(K))})$ for K sufficiently large. This means that there exists constants $\gamma 1 > 0$ and $\gamma 2 > 0$ and a macro iteration K_0 such that $\gamma 1\, e^{-(d^*/t(K))} \leq r(K,G) \leq \gamma 2\, e^{-(d^*/t(K))}$ for all $K \geq K_0$. Therefore, for all $K \geq K_0$,

$$\prod_{k=1}^{K} (1 - \gamma 2\, e^{-(d^*/t(k))}) \leq P\{B_{SA}(K,G)\} = \prod_{k=1}^{K} [1 - r(k,G)] \leq \prod_{k=1}^{K} (1 - \gamma 1\, e^{-(d^*/t(k))}). \qquad (11)$$

Hajek's condition on the cooling schedule (and indirectly, through the choice of neighborhood function which defines d*), $\sum_{K=1}^{+\infty} e^{-(d^*/t(K))} = +\infty$, places restrictions on the rate at which the cooling schedule t(k)

approaches zero. Suppose that the cooling schedule is defined such that $e^{-(d^*/t(K))} = \lambda(1/k)^{\delta}$ for $k \geq 2$ and $\lambda \in Z^+$, for some $0 < \delta \leq 1$. Note that t(k) could also be define using iterated logarithms (e.g., $e^{-(d^*/t(K))} = \lambda(1/k(\ln(k))$ or any other form provided that Hajek's condition is satisfied.

From the expression in (11), one can obtain an upper and lower bound on $e^{\varphi K} P\{B_{SA}(K,G)\}$ as $K \rightarrow +\infty$. In particular, for all $K \geq K_0$,

$$e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 2\, e^{-(d^*/t(k))}) \leq e^{\varphi K} P\{B_{SA}(K,G)\} \leq e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 1\, e^{-(d^*/t(k))})$$

which leads to

$$e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 2\, \lambda(1/k)^{\delta}) \leq e^{\varphi K} P\{B_{SA}(K,G)\} \leq e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 1\, \lambda(1/k)^{\delta}). \tag{12}$$

Since $\prod_{k=1}^{+\infty} (1 - \gamma 1\, \lambda(1/k)^{\delta}) = 0$ if and only if $\sum_{K=1}^{+\infty} (1/k)^{\delta} = +\infty$, then the rate at which $\prod_{k=1}^{K} (1 - \gamma 1\, \lambda(1/k)^{\delta})$ approach zero as $K \rightarrow +\infty$ relative to the rate at which $e^{\varphi K}$ approaches infinity determines which of the four cases described in Theorem 11 apply to this simulated annealing algorithm.

To determine this rate, note that for some $k0 \in Z+$ where $\gamma 2\, \lambda(1/k0)^{\delta} < 1$,

$$\ln\left(\prod_{k=k0}^{K} (1 - \gamma 1\, \lambda(1/k)^{\delta})\right) = \sum_{k=k0}^{K} \ln(1 - \gamma 1\, \lambda(1/k)^{\delta}) \leq \sum_{k=k0}^{K} -\gamma 1\, \lambda(1/k)^{\delta}. \tag{13}$$

Using the integral approximation for (14), this expression is $O(-\ln(K))$ as $K \rightarrow +\infty$ for $\delta=1$, and $O(-K^{1-\delta})$ as $K \rightarrow +\infty$ for $0<\delta<1$. By taking the exponential function in (14), then $\prod_{k=1}^{K} (1 - \gamma 1\, \lambda(1/k)^{\delta})$ is $O(1/K)$ as $K \rightarrow +\infty$ for $\delta=1$, and $O(\exp(K^{\delta-1}))$ as $K \rightarrow +\infty$ for $0<\delta<1$. Therefore, $e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 1\, \lambda(1/k)^{\delta}) \rightarrow +\infty$ as $K \rightarrow +\infty$ for $0<\delta\leq 1$. The same conclusions are obtained from the lower bound in (12). In particular,

$$\ln\left(\prod_{k=k0}^{K} (1 - \gamma 2\, \lambda(1/k)^{\delta})\right) = \sum_{k=k0}^{K} \ln(1 - \gamma 2\, \lambda(1/k)^{\delta}) \leq \sum_{k=k0}^{K} -\gamma 2\, \lambda(1/k)^{\delta} / (1 - \gamma 2\, \lambda(1/k)^{\delta}). \tag{14}$$

Once again, using the integral approximation for (14), this analysis yields the same results. Therefore, $e^{\varphi K} \prod_{k=1}^{K} (1 - \gamma 2\, \lambda(1/k)^{\delta}) \rightarrow +\infty$ as $K \rightarrow +\infty$ for $0<\delta\leq 1$. This means that conditions i), ii), and iii) cannot hold for this simulated annealing algorithm. Therefore, either condition iv) holds (provided $\varphi \leq \varphi_0$), hence random restart local search dominates this simulated annealing algorithm, or the results are inconclusive (if $\varphi > \varphi_0$). Note that if φ_0 is very close to zero, hence P{L-stop} is very close to one, then if condition iv) holds, the value for K_0 may be prohibitively large, by comparing $e^{\varphi K}$ with functions that are $O(1/K)$ or

$O(\exp(K^{\delta-1})$. Therefore, the form of the cooling schedule for simulated annealing and the choice of neighborhood function that defines the value for P{L-stop} for random restart local search determine the relative performance of these algorithms. To make this result more practical, it is necessary to determine the number of restarts/macro iterations K_0 that are needed such that $P\{(B_{RR}(K,G))^c\} \geq P\{(B_A(K,G))^c\}$ for all $K \geq K_0$; this is a topic of current research and investigation.

Note that the vast majority of theoretical results on simulated annealing are concerned with its asymptotic convergence. The results in Theorems 10 and 11 suggest that random restart local search will outperform simulated annealing provided that a sufficiently large number of restarts are executed. The primary value of using simulated annealing may therefore be for *finite-time* executions that obtain near-optimal solutions reasonably quickly. This, in turn, suggests that one should focus on the finite-time behavior of simulated annealing rather than the asymptotic convergence results that dominate the literature. These results also imply that the primary value of random restart local search is when it is applied over an infinite horizon. Work is in progress to use and extend these results to identify both convergent and nonconvergent GHC algorithm formulations that perform well over finite horizons, as well as determine the number of random restarts that are needed to satisfy the inequalities in Theorems 10 and 11. Moreover, work is in progress to determine how the framework provided by the micro/macro iteration structure can be further exploited to gain insights into the finite-time and asymptotic performance of GHC algorithms in general.

5. Applications

This section describes results in applying generalized hill climbing algorithms to manufacturing process design optimization problems (Fischer et al. 1997) of interest to the Air Force. This effort has been interdisciplinary, involving researchers from the Materials Process Design Branch of the Air Force Research Laboratory (Wright Patterson Air Force Base), and Austral Engineering and Software, Inc., (Athens, Ohio). Note that. The research and algorithm development work by our research group has been transitioned to Austral Engineering and Software, Inc., hence supporting their effort under a Phase II SBIR through the Air Force. This collaboration between an Air Force Laboratory, and industrial partner, and an academic institution has been highly productive and extremely synergistic in identifying problems of interest to the Air Force, as well as formulating and developing solutions to address such problems.

An important applied development has been the formulation of a hybrid algorithm that crosses ordinal optimization with generalized hill climbing algorithms. These results are reported in Sullivan and Jacobson (2000b). Ordinal optimization (Ho et al. 1992) is a search procedure that can be used to obtain optimal/near-optimal designs for discrete manufacturing process design optimization problems. The strength of the ordinal optimization approach is in its focus on finding good designs, rather than trying to

find the very best design (i.e., goal softening) (Lee et al. 1998). This allows ordinal optimization to reduce sampling over a very large design space (when searching for an optimal design) to sampling over a smaller, more manageable, set of good designs (see Chen and Kumar 1996 for an application of ordinal optimization to a discrete optimization problem). The strength of the generalized hill climbing algorithm is in its intelligent exploration of the design space when searching for an optimal design. In practice, generalized hill climbing algorithms find good designs in finite time, similar to ordinal optimization.

Ordinal Optimization is an algorithmic approach for identifying good designs over a large design space. By relaxing the requirement to obtain the optimal design, hence searching for designs that are in a small percentage of the best designs, the algorithm can efficiently weed out very poor designs and identify a set of designs (i.e., the selected subset) among which there exists a good design (i.e., an element of the good enough subset). The quality of the selected subset with respect to the good enough subset is obtained using an *alignment probability*, which measures the probability that these two subsets contain some integer value number of overlapping (common) designs. The very nature of relaxing the optimization requirement in this way has suggested the term *soft optimization* as appropriate for describing the execution of ordinal optimization.

By relaxing (or softening) the optimization requirement for ordinal optimization, a fast convergence rate for the algorithm can be obtained. In particular, Dai (1996) proves that the ordinal optimization convergence rate is exponential in the number of designs selected. He shows this by considering a set of t designs, selected independently and identically over the entire design space, and defining the *correct selection* (CS) event as the event in which the best design (over the t selected) is indeed the best overall design. Then, under mild restrictions, P{CS} is shown to be bounded below by $1-\alpha e^{-\beta t}$, where α and β are positive constants. Convergence can also established by showing that the alignment probability converges to zero. These results show that relaxing an algorithm's requirement from finding the overall best design, to finding a design in some top percentile of all designs in the design space, results in a significantly improved rate of convergence.

Using the strengths of both ordinal optimization and generalized hill climbing algorithms, a new algorithm is proposed, termed *ordinal hill climbing*. The ordinal hill climbing algorithm is a particular type of generalized hill climbing algorithm, with the hill climbing component of the generalized hill climbing algorithm based on ordinal information, where a set of designs are collected and evaluated at each iteration (similar to how ordinal optimization is applied). In particular, the algorithm is initialized with a set (of M) selected designs, chosen among all the designs in the design space (e.g., randomly). At each iteration k, a hill climbing random variable, R_k, defined over the discrete values {1, 2, ...,M} is generated ($R_k = r_k$) and used to keep the best r_k designs in the selected set, where the designs in the

selected set are ordered from smallest to largest cost, with best referring to the designs with smallest cost. The $M-r_k$ open spots in the selected set are then filled with new designs, chosen among all the designs in the design space, or using some rule based on the designs that were kept in the selected set. This process is iteratively repeated until the algorithm terminates (e.g., a fixed number of iterations are executed). The ordinal hill climbing algorithm is described in pseudo-code form, given a design space, Ω, and a cost function (f) and a neighborhood function η (if needed) defined over the design space.

Select a set of M initial designs $D(0) \subset \Omega$
Set the iteration number $k = 0$
Define the hill climbing variable R_k, where
    i) $\Sigma^{M}_{m=1} P\{R_k = m\} = 1$, *ii)* $R_0 = 1$ with probability one,
Repeat
    Order the designs in $D(k)$ from smallest to largest cost function values
    Generate R_k $(= r_k)$
    Keep the best (smallest cost function value) r_k designs from the set $D(k)$. Call this set E_1.
    Generate $M-R_k$ new designs from the design space Ω. Call this set E_2.
    Set $D(k+1) \leftarrow E_1 \cup E_2$.
    $k \leftarrow k+1$
Until stopping criterion is met

There are several parameters that must be initialized for the ordinal hill climbing algorithm. First, the value of M, the size of the selected design sets, *D(k),* must be set. Second, the initial set of M manufacturing process designs, $D(0)$, can be selected randomly or using any specified selection rule. In addition, there are two features of the algorithm where the user has great flexibility. First, the hill climbing variable R_k can be defined as any discrete (random or deterministic) non-negative variable defined over the set of integers (from one to M). Second, at each iteration, the method of filling in the $M- r_k$ designs in the selected set must be specified (e.g., generate neighbors of each of the designs that are kept in the selected set).

The intuition behind ordinal hill climbing algorithms is that at each iteration, the best designs, where best is measured using the value of the cost function (smaller cost designs are better than larger cost designs), are more likely to become part of, and remain in, the selected design set $D(k)$. The hill climbing variable determines the acceptance or rejection of designs in the design set based on their relative costs (i.e., ordinal rank). Therefore, the *ranking* of the design costs, rather than the costs themselves, determines whether a design is accepted. For example, if R_k is close to one with high probability, as in the early stages of the algorithm execution, then most of the designs will be rejected from the selected set during this phase, hence guaranteeing a broad (and aggressive) sampling of designs.

On the other hand, if as the algorithm progresses R_k is more likely to be close to M, then the algorithm will tend to retain good designs that have managed to remain in the selected design set.

In light of this observation, a possible third requirement on the hill climbing variable is $R_k \rightarrow M$ (with probability one) as $k \rightarrow +\infty$. As the algorithm proceeds, this requirement ensures that the algorithm will accept an increasing number of designs, until at termination, when $P\{\lim_{k\rightarrow+\infty} R_k = M\} = 1$, the algorithm accepts all of the designs in the selected design set. The defining feature of the ordinal hill climbing algorithm is that at termination, the final set of designs is likely to contain a good design. To identify a subset of this final set of designs to use, ordinal optimization can be applied.

There are a number of ways to simplify and/or fine-tune the ordinal hill climbing algorithm. If M is set to one at all iterations, then each iteration considers only a single design. This simplification reduces the ordinal hill climbing algorithm to a generalized hill climbing algorithm. There are an unlimited number of choices for the hill climbing variable, R_k, that satisfy the requirements listed in the ordinal hill climbing pseudo-code. There are also numerous ways in which the selected design set can be updated from $D(k)$ to $D(k+1)$, depending, for example, on the choice of neighborhood function. To illustrate, this update can be done using multiple neighbors of only the best, or near-best (e.g., top $\beta * M$, where $\beta > 0$ close to zero) designs. Stopping criterion that can be used for the ordinal hill climbing algorithm include looking at the top αM designs, with $\alpha > 0$ close to zero, and stopping the algorithm if this top αM set of designs does not change over some specified number of iterations. Each of these modifications results in a large selection of ordinal hill climbing algorithm variations that can be used to address the an integrated blade rotor discrete manufacturing process design optimization problem of interest to the Air Force. Computational results with the ordinal hill climbing algorithm applied to this problem are reported in Sullivan and Jacobson (2000b).

The relationship between genetic algorithms and ordinal hill climbing algorithms was also studied. Genetic algorithms (GA) are an optimization strategy that has been successfully applied to numerous discrete optimization problems. The foundation of GA is derived from the Darwinian notion of "survival of the fittest" (Reeves 1993); this notion suggests that as time evolves, parents produce new offspring that are more acceptable than members of previous populations. In other words, over a period of time, parents are continually mating to produce successive generations of children that are closer to optimality than their predecessors.

The GA ordinal component of comparing cost function values over a set of children (i.e., designs) suggest that there may be a natural bridge between GA and ordinal hill climbing algorithms. Though GA have proven to be effective for addressing intractable discrete optimization problems, and can be classified as a type of hill climbing approach, its link with generalized hill climbing algorithms

(through the ordinal hill climbing formulation) provides a well-defined relationship with other generalized hill climbing algorithms (like simulated annealing and threshold accepting). Therefore, such an analysis provides useful insights and observations that may fuel further research into both ordinal hill climbing and genetic algorithms, and how they fit together on a broader scale.

The bridge between GA and the ordinal hill climbing algorithm framework is defined through the hill climbing variable R_k and the method by which each successive selected subset is updated (based on a neighborhood function definitions). In particular, the hill climbing variable R_k for ordinal hill climbing algorithms determines the number of parents carried over from the selected set (population) $D(k)$ to the set E_1. Once the set E_1 has been determined, its members can be mated (e.g., according to a crossover rule) to produce $M-R_k$ offspring. Therefore, the GA concept of mating serves as the neighborhood function in the ordinal hill climbing algorithm framework. This set of $M-R_k$ offspring, E_2, together with the set E_1 becomes the next selected set or population (i.e., $D(k+1) = E_1 \cup E_2$).

Basic GA consist of three components:

i) Reproduction
ii) Crossover
iii) Mutation

Reproduction is the process by which individual parents are evaluated for mating and inclusion in future populations. Variants of GA can be described by employing different hill climbing variables R_k (i.e., the selection of parents to be contained in E_1 varies with the choice of R_k). Each parent in a population is evaluated according to its cost function value. In general, parents with good (lower) cost function values are more likely to be among the R_k selected parents for the set E_1. However, for diversification, the set E_1 can be a mix of both good and bad parents. *Crossover* is a process by which offspring are generated based upon the cost function values of the parents. A particular crossover method needs to be defined before the GA is executed. Crossover may consist of generating a neighbor of a parent in E_1 to produce offspring or combining different parts of two parents in E_1 to produce offspring.

The following is the ordinal hill climbing algorithm formulation of *simple* GA, which uses reproduction and crossover components:

Select a set of M initial designs $D(0) \subset \Omega$
Set the iteration number k = 0
Repeat
  Order the designs in D(k) from smallest to largest cost function values
  Generate R_k (= r_k)
  Keep the best (smallest cost function value) r_k designs from the set D(k) as parents for creating offspring. Call this set E_1.
  Generate offspring of E_1 according to the crossover rule to obtain $M-r_k$ new designs. Call this set E_2.
  Set D(k+1) $\leftarrow E_1 \cup E_2$.

$k \leftarrow k+1$
Until stopping criterion is met

Mutation involves the random alteration of a parameter value in the offspring. Mutation can play either a primary or secondary role in GA, depending on the desired aggressiveness of GA. If the entire population has cost function values that are relatively close to each other, the search may easily get caught at a local optimum. In this situation, it may be desirable to increase the probability of mutation to ensure the inclusion of designs in the set E_1 sufficiently different from those already in the population.

The following is the ordinal hill climbing formulated simple GA with a mutation component.

Select a set of M initial designs $D(0) \subset \Omega$
Set the mutation probability
Set the iteration number $k = 0$
Repeat
  Order the designs in $D(k)$ from smallest to largest cost function values
  Generate R_k $(= r_k)$
  Keep the best (smallest cost function value) r_k designs from the set $D(k)$ as parents for creating offspring. If the largest and smallest cost function values of parents among these r_k values are relatively close to each other, increase the mutation probability, hence the diversity of the parents set. Call this set E_1.
  Generate offspring of E_1 according to the crossover rule to obtain $M-r_k$ new designs. Call this set E_2.
  Set $D(k+1) \leftarrow E_1 \cup E_2$.
  $k \leftarrow k+1$
Until stopping criterion is met

At the end of each iteration, the new population becomes the population of parents that will be used to perform reproduction and crossover to generate yet another new population. This process is repeated until the algorithm terminates; at termination, the final population will (hopefully) contain an optimal parent (i.e., an optimal design).

These ordinal hill climbing algorithm formulations for GA illustrate the power and flexibility of the ordinal hill climbing algorithm framework. It should be noted that by defining the procedure by which the selected subset in ordinal hill climbing algorithms is updated without a neighborhood function, it is possible to obtain an ordinal hill climbing algorithm formulation that is not a GA (such as by simply randomly generating designs to obtain set E_2). Moreover, problem-specific implementations for GA may not fit into the ordinal hill climbing framework. Further research is needed to fully assess the relationship between ordinal hill climbing algorithms and GA. However, the two formulations presented provide a first step towards developing a bridge between GA and other search strategies like simulated annealing, threshold accepting, and tabu search (Johnson 1996) using the generalized hill climbing paradigm (Johnson and Jacobson 2001a,b). Moreover, these formulations serve to illustrate the power of the ordinal optimization strategy in addressing deterministic discrete optimization problems.

Lastly, several on-site meetings and interactions with Austral Engineering and Software, Inc., has resulted in further enhancement to the generalized hill climbing algorithm framework and software that our research group has developed for their use. Moreover, research on optimization across several manufacturing process design optimization sequence (see Vaughan et al. 2000) has resulted in a new neighborhood function definition and form that has the potential to support the development of automated optimization procedures for such problems. This on-going effort of developing and enhancing the generalized hill climbing algorithm code, in conjunction with the software development efforts of Austral Engineering and Software, Inc., as well as the theoretical work being undertaken to better understand the generalized hill climbing algorithm framework and how to exploit its form to improve performance, all serve as a powerful mechanism for transitioning this research from an academic environment into an industrial setting.

6. Other Research Results

In addition to the results reported with generalized hill climbing algorithms and discrete manufacturing process design optimization problems, several other results were obtained during the period of the grant. These results are briefly discussed here.

Discrete optimization and discrete event simulation are two disparate areas within the general field of operations research. Building a framework that facilitates the cross-fertilization of tools in these two areas has provided much of the motivation and basic research leading up to the creation of the generalized hill climbing algorithm paradigm. Jacobson and Yucesan (1999) formulate four search problem for discrete event simulation models and prove them to be NP-hard. These problems consider fundamental issues faced by simulation practitioners (e.g., can a state be accessed, does changing the order of events impact the states, does representing the same simulation model on a computer in two different ways result in different states when executed, can a simulation run stall, how do permutations of events impact the states and future events lists, or how events are cancelled, resulting in a loss of information). The first problem, termed ACCESSIBILITY, is a generalization of the discrete manufacturing design problems briefly described in Section 5.

Fleischer and Jacobson (1999) consider how information theory can be used to assess the finite-time performance of the simulated annealing algorithm. These results provide insights into how simulated annealing can be executed to optimize its effectiveness. Jacobson and Schruben (1999) look at how harmonic analysis can be used to perform sensitivity analysis of steady state discrete event simulation models. Kumar et al. (2000) present an analysis of a scheduling optimization problem associated with flexible assembly systems. Jacobson et al. (2000, 2001) analyze access control security systems using mathematical programming techniques and formulations. Jacobson and Morrice (1998)

study a combinatorial medical problem that evaluates the temporal association between health disorders and medical treatments.

REFERENCES

Chen, C.H., Kumar, V., 1996, "Motion Planning of Walking Robots in Environments with Uncertainty," *Proceedings of IEEE Conference on Robotics and Automation.*

Dai, L., 1996, "Convergence Properties of Ordinal Comparison in the Simulation of Discrete Event Dynamic Systems, " *Journal of Optimization Theory and Application*, 91(2), 363-388.

Fischer, C.E., Gunasekera, J.S., Malas, J.C., 1997, "Process Model Development for Optimization of Forged Disk Manufacturing Processes," *Steel Forgings*, Second Volume, ASTM STP 1257, E.G. Nisbett and A.S. Melilli, Editors, American Society for Testing and Materials.

Fleischer, M.A., Jacobson, S.H., 1999, "Information Theory and the Finite-Time Behavior of the Simulated Annealing Algorithm: Experimental Results," *INFORMS Journal on Computing*, 11(1), 35-43.

Hajek, B., 1988, "Cooling Schedules for Optimal Annealing," *Mathematics of Operations Research,* 13, 311-329.

Ho, Y.C., Sreenivas, R., Vakili, P., 1992, "Ordinal Optimization of Discrete Event Dynamic Systems," *Journal of Discrete Event Dynamical Systems,* 2(2), 61-88.

Jacobson, S.H., Kobza, J.E., Nakayama, M.K., 2000, "A Sampling Procedure to Estimate Risk Probabilities in Access Control Security Systems," *European Journal of Operational Research,* 122(1), 123-132.

Jacobson, S.H., Kobza, J.E., Simms, A.E., 2001, "A Detection Theoretic Approach to Modeling Aviation Security Problems Using the Knapsack Problem," *IIE Transactions.*

Jacobson, S.H., Morrice, D.J., 1998, "A Mathematical Model for Assessing the Temporal Association Between Health Disorders and Medical Treatments," *Journal of Statistical Planning and Inference*, 71(1/2), 209-228.

Jacobson, S.H., Schruben, L.W., 1999, "A Harmonic Analysis Approach to Simulation Sensitivity Analysis," *IIE Transactions*, 31(3), 231-243.

Jacobson, S.H., Solow, D., 1993, "The Effectiveness of Finite Improvement Algorithms for Finding Global Optima," *Zeitschrift fur Operations Research (ZOR) -- Methods and Models of Operations Research*, 37(3), 257-272.

Jacobson, S.H., Sullivan, K.A., Johnson, A.W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

Jacobson, S.H., Yucesan, E., 1999, "On the Complexity of Verifying Structural Properties of Discrete Event Simulation Models," *Operations Research*, 47(3), 476-481.

Jacobson, S.H., Yucesan, E., 2000a, "On the Effectiveness of Generalized Hill Climbing Algorithms," Technical Report, University of Illinois, Urbana, Illinois.

Jacobson, S.H., Yucesan, E., 2000b, "Assessing the Performance of Generalized Hill Climbing Algorithms," Technical Report, University of Illinois, Urbana, Illinois.

Johnson, A.W., 1996, "Generalized Hill Climbing Algorithms for Discrete Optimization Problems," Ph.D. Dissertation, Department of Industrial and Systems Eng., Virginia Tech, Blacksburg, Virginia.

Johnson, A.W., Jacobson, S.H., 2001a, "A Convergence Result for a Class of Generalized Hill Climbing Algorithms," *Applied Mathematics and Computation* (Accepted).

Johnson, A.W., Jacobson, S.H., 2001b, "A General Convergence Result for Hill Climbing Algorithms," *Discrete Applied Mathematics* (Accepted).

Kumar, A., Jacobson, S.H., Sewell, E.C., 2000, "Computational Analysis of a Flexible Assembly System Design Problem," *European Journal of Operational Research*, 123(3), 17-36.

Lee, L.H, Lau, T.W.E., Ho, Y.C. 1998, "Explanation of Goal Softening in Ordinal Optimization," *IEEE Transactions on Automatic Control.*

Reeves, C.R., 1993, *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, Inc., New York, New York.

Ross, S.M., 1988, *A First Course in Probability*, Third Edition, Macmillan Publishing Co., New York.

Sullivan, K.A., 1999, "A Convergence Analysis of Generalized Hill Climbing Algorithms," Ph.D. Dissertation, Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, Virginia.

Sullivan, K.A., Jacobson, S.H., 2000a, "A Convergence Analysis of Generalized Hill Climbing Algorithms," Technical Report, University of Illinois, Urbana, Illinois.

Sullivan, K.A., Jacobson, S.H., 2000b, "Ordinal Hill Climbing Algorithms for Discrete Manufacturing Process Design Optimization Problems," *Discrete Event Dynamical Systems* 10(4), 307-324

Vaughan, D.E., Jacobson, S.H., Armstrong, D.E., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," *ASME Journal of Mechanical Design*, 122(2), 164-171.

CONTRIBUTING PERSONNEL

The principal investigator for this project, Dr. Sheldon H. Jacobson, has devoted 33% of his academic year time, and 33% of his summer, in each of the three years of this project. Major Alan W. Johnson (US Air Force) has provided input on this project throughout the three years of the grant. Dr. Kelly Ann Sullivan and Dr. Diane E. Vaughan were two Ph.D. students of the principal investigator, who worked on various aspects of this project. Lieutenant Colonel Darrell Henderson (US Army), Mr. Derek Armstrong, Mr. Jon Bowman, and Mr. Jeffrey Orosz are current graduate students of the principal investigator who have also contributed to various aspects of this project.

MEDIA COVERAGE

The paper "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," (co-authored with Diane E. Vaughan and Derek E. Armstrong) published in *ASME Journal of Mechanical Design* (Volume 122, Number 2, 164-171) was featured in an article on October 20, 2000 focusing on new directions in manufacturing research in *Advanced Manufacturing Technology* (www.wiley.com/technical_insights).

TRANSITIONS

Extensive interactions with Austral Engineering and Software, Incorporated, have resulted in generalized hill climbing algorithm commercial quality software code. This code is being developed as library functions that are compatible with the interactive framework (MPDX©) being developed by Austral Engineering and Software, Incorporated, as part of their SBIR Phase II contract through Wright Patterson Air Force Base.

PUBLICATIONS

The following is a list of publications that have resulted from research on this grant.

Submitted but not yet Accepted

Sullivan, K.A., Jacobson, S.H., 2000, "A Convergence Analysis of Generalized Hill Climbing Algorithms."

Jacobson, S.H., Yucesan, E., 2000, "On the Effectiveness of Generalized Hill Climbing Algorithms."

Jacobson, S.H., Yucesan, E., 2000, "Assessing the Performance of Generalized Hill Climbing Algorithms."

Published or Accepted

Fleischer, M.A., Jacobson, S.H., 1999, "Information Theory and the Finite-Time Behavior of the Simulated Annealing Algorithm: Experimental Results," *INFORMS Journal on Computing*, 11(1), 35-43.

Jacobson, S.H., Kobza, J.E., Nakayama, M.K., 2000, "A Sampling Procedure to Estimate Risk Probabilities in Access Control Security Systems," *European Journal of Operational Research*, 122(1), 123-132.

Jacobson, S.H., Kobza, J.E., Simms, A.E., 2001, "A Detection Theoretic Approach to Modeling Aviation Security Problems Using the Knapsack Problem," *IIE Transactions* (Accepted).

Jacobson, S.H., Morrice, D.J., 1998, "A Mathematical Model for Assessing the Temporal Association Between Health Disorders and Medical Treatments," *Journal of Statistical Planning and Inference*, 71(1/2), 209-228.

Jacobson, S.H., Schruben, L.W., 1999, "A Harmonic Analysis Approach to Simulation Sensitivity Analysis," *IIE Transactions*, 31(3), 231-243.

Jacobson, S.H., Sullivan, K.A., Johnson, A.W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

Jacobson, S.H., Yucesan, E., 1999, "On the Complexity of Verifying Structural Properties of Discrete Event Simulation Models," *Operations Research*, 47(3), 476-481.

Johnson, A.W., Jacobson, S.H., 2001, "A Class of Convergent Generalized Hill Climbing Algorithms," *Applied Mathematics and Computation* (Accepted).

Johnson, A.W., Jacobson, S.H., 2001, "On the Convergence of Generalized Hill Climbing Algorithms," *Discrete Applied Mathematics* (Accepted).

Kumar, A., Jacobson, S.H., Sewell, E.C., 2000, "Computational Analysis of a Flexible Assembly System Design Problem," *European Journal of Operational Research*, 123(3), 453-472.

Sullivan, K.A., Jacobson, S.H., 2000, "Ordinal Hill Climbing Algorithms for Discrete Manufacturing Process Design Optimization Problems," *Discrete Event Dynamical Systems*, 10(4), 307-324

Vaughan, D., Jacobson, S.H., Armstrong, D.E., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization Using Generalized Hill Climbing Algorithms," *ASME Journal of Mechanical Design*, 122(2), 164-171.